Mechanisms for decentralized online scheduling

Mechanisms for Decentralized Online Scheduling[†]

Birgit Heydenreich[*] Rudolf Müller[*] Marc Uetz[*]

July 6, 2005

Abstract

The paper introduces a model for online parallel machine scheduling, where any single machine is run on the basis of a locally optimal sequencing policy. Jobs choose the machine on which they want to be processed themselves, and in addition, any job j owns a piece of private information, namely its indifference cost w_j for waiting one additional unit of time before being processed. We study this setting from the perspective of algorithmic mechanism design, and assuming that each job prefers to be completed as early as possible, the utilitarian social choice function minimizes the total weighted completion times $\sum w_j\, c_j$. We prove that in this setting there exists an online mechanism, running in polynomial time, where rational jobs select their machine in such a way that the resulting schedule is 3.281-competitive with respect to the off-line optimal solution that maximizes social welfare. The mechanism deploys an online payment scheme that induces rational jobs to truthfully report their indifference costs, in the sense that it is a myopic best response. Moreover, the payment scheme results in a balanced budget, that is, payments are only made between jobs. We also discuss extensions to mechanisms where truth-telling is even an ex-post weakly dominant strategy, while preserving the competitive ratio.

1 Introduction

We study the online version of the classical parallel machine scheduling problem to minimize the total weighted completion time –P $\mid r_j \mid \sum w_j\, c_j$ in the notation of Graham et al. [6]– from a new perspective: We assume that the system needs to be organized without (too much of) central coordination. More precisely, we ask for the performance of the system if each of the parallel machines is run on the basis of a reasonable (yet suboptimal) sequencing policy locally, while the online arriving jobs have to decide for the machines themselves. Each arriving job would like to be scheduled as early as possible, and it comes with a piece of information that is not publicly known, namely its indifference cost w_j for waiting one unit of time. The processing times p_j of jobs are assumed to be publicly known. The indifference costs w_j, together with the processing times p_j of the jobs, however, define the input for the local scheduling policies of the machines. An arriving job may thus have an incentive to lie about its true indifference cost, in order to strategically manipulate the schedule. The goal is to nevertheless set up a system that yields a reasonable overall performance. In this setting, it is typically desirable to implement a social choice function that is utilitarian, i.e., it maximizes the sum of the valuations of all the jobs [17]. Hence, since any job prefers to be completed as early as possible, with indifference cost w_j for waiting, a mechanism

[*]Maastricht University, Quantitative Economics, P.O. Box 616, 6200 MD Maastricht, The Netherlands. E-mail: {b.heydenreich,r.muller,m.uetz}@ke.unimaas.nl

[†]This research was supported by NWO grant 2004/03545/MaGW 'Local Decisions in Decentralised Planning Environments'.

that minimizes the total weighted completion time $\sum w_j c_j$ implements the utilitarian social choice function, or in other words, it maximizes the total social welfare.

From a pure algorithmic perspective, this problem poses two challenges, as stated also by Koutsoupias and Papadimitriou [11]. The first is the *lack of information* caused by the fact that jobs become known to the system only at the moment of their arrival (online algorithms). The second is the *lack of unbounded computational resources*, which allows only for approximate solutions to the underlying NP-hard optimization problem (approximation algorithms). A third obstruction, in fact the starting point for what is known as *mechanism design*, is the *asymmetry of information*: information that is required to run the system is private to the participants in the system. Selfish agents trying to maximize their own benefit can therefore do so by reporting strategically about their private information, thus manipulating the resulting outcome.

Contribution. The paper touches on three research areas, namely the design of approximation algorithms for NP-hard optimization problems, competitive analysis for online optimization, and mechanism design for resolving the asymmetry of information. Motivated by the observation that many real-life systems are not centrally but rather hierarchical organized, we assume that any of the parallel machines adopts a 'reasonable' scheduling policy locally without further reference to what is happening on other machines. More precisely, we assume that any machine locally utilizes the well known WSPT rule, since this is the locally optimal scheduling policy for the single machine scheduling problem $1||\sum w_j c_j$ [20]. That is, jobs are scheduled in the order of relative shortest processing time p_j/w_j first. Intuitively, the computational complexity of parallel machine scheduling is thus bypassed, of course at a possible loss in overall performance. Moreover, we assume that the WSPT rule is used in a myopic, yet practically reasonable sense, namely without further consideration of the online situation. So at any time, a machine schedules among all jobs assigned to it the one with largest ratio w_j/p_j first. Clearly, some procedure is needed to coordinate the distribution of jobs over machines. Moreover, given that the knowledge of the private information of the jobs, their indifference costs w_j, is vital to achieve a reasonable overall performance, and given that any job selfishly seeks to minimize its own completion time, we need to set up a mechanism that induces them to choose the 'right' machine, and to report their private information w_j truthfully to the respective machine.

We prove in this paper that such a mechanism exists. More precisely, we present a polynomial-time mechanism for the previously described model for decentralized online scheduling. As usual in mechanism design, our mechanism defines *payments* that have to be made by the jobs, depending on the machine they choose, and depending on the report of their private information w_j. These payments are used for two different purposes. First, as a means of coordination, inducing the jobs to choose the 'right' machine, and second, as a means to stimulate truthful reports of the w_j's. Building upon an analysis of Megow et al. [16], we show that the so-defined decentralized mechanism is 3.281-competitive, that is, under the assumption of rational behavior of the jobs, the obtained solution is never more than 3.281 times the optimal off-line solution that uses central coordination. Note that the performance bound matches the one obtained by Megow and Schulz [15] for the online parallel machine problem to minimize $\sum w_j c_j$ (using central coordination). Moreover, the payments result in a balanced budget, that is, no payments are required from or to a central coordination authority. In the mechanism, a truthful report of the indifference cost w_j is a myopic best response for any job. We also give an example to show that no payments exist that would make truth-telling an ex-post weakly dominant strategy. In other words, the algorithm cannot be turned into a dominant strategy incentive compatible mechanism. However, we show that dominant strategy incentive compatible mechanisms can be obtained, while maintaining the same performance bound, at the cost of introducing central coordination for the distribution of the jobs

over machines.

We see the contribution of the paper twofold. First, we introduce a natural paradigm for decentralization of a parallel machine system: Each machine implements a locally optimal sequencing policy, while jobs have to choose the machines themselves. Moreover, the processing requirements p_j of the jobs are assumed to be publicly known, since they are publicly observable once the jobs are scheduled. Private information is only the indifference cost w_j of any job, measuring the private cost for being processed one unit of time later. Second, we show that in this decentralized model, both the lack of central coordination and the asymmetry of information can be bypassed by the definition of very simple and natural payments that even result in a balanced budget, at least assuming a certain notion of rational behavior. Most importantly, of course, the resulting mechanism for the online parallel machine scheduling problem provides a competitive ratio of 3.281.

Related Work. Mechanism design in combination with the design of approximation algorithms for scheduling problems has been studied, e.g., by Nisan and Ronen [18] and Archer and Tardos [2]. In both papers, not the jobs but the machines are the selfishly behaving parts of the system, and the private information is the processing speed of the machine. Archer and Tardos [2] consider as application of their main theorem a problem with *related* parallel machines, that is, each machine i has a processing speed s_i, and a job j requires processing for p_j/s_i time if processed on machine i. They define a necessary and sufficient condition for the existence of truthful (dominant strategy incentive compatible) mechanisms, namely the monotonicity of the total work assigned to a machine in dependence on its reported speed s_i. On the basis of this characterization, they design a randomized mechanism that yields a deterministic 3-approximation for the minimum makespan problem, $\mathrm{Q}\,|\,|\,C_{\max}$, while truthfully reporting s_i is a dominant strategy that maximizes the expected utility of each machine. A deterministic monotone algorithm with the same performance bound was recently proposed by Kovacs [12]. Nisan and Ronen [18] consider the minimum makespan problem on unrelated machines, $\mathrm{R}\,|\,|\,C_{\max}$. There, the processing time of any job j on any machine i is p_{ij}, with no relation to speeds of the machines. In their paper, the private information of any machine i is multi-dimensional, namely the vector of processing times $(p_{i1}, \ldots, p_{in})$. They present a truthful (dominant strategy incentive compatible) m-approximation, where m denotes the number of machines. Furthermore, they prove that no truthful mechanism can yield a ϱ-approximation for the minimum makespan problem with $\varrho < 2$, and present a randomized mechanism that beats this deterministic lower bound. Decentralized scheduling is also subject of a paper by Wellman et al. [22]. In contrast to the model we propose, however, the term 'decentralized' in [22] describes the fact that agents choose their bidding strategy on the basis of local information, which differs from our notion of decentralization.

In the sequel of the paper, we formalize the model and introduce the required notation in Section2. In Section3 we first give a simple example to show that decentralization may be arbitrarily bad in general, and propose an algorithm for online scheduling on parallel machines using central coordination, inspired by [16]. In Section 4, we analyze this algorithm from a mechanism design perspective, and in Section 5 we introduce a simple payment scheme that yields a decentralized implementation of the same algorithm, in such a way that truthful reporting the indifference costs w_j is a myopic best response for any job. We analyze the performance of the resulting mechanism in Section 6 and conclude with further extensions and remarks in Section 7.

2 Model and Notation

We consider the online scheduling problem with non-trivial release dates on parallel machines with the objective to minimize the weighted sum of completion times, $\mathrm{P}\,|\,r_j\,|\,\sum w_j\,c_j$. We are given a

set of jobs $J = \{1, \ldots, n\}$, with processing requirements $p_j > 0$, $j \in J$, and each job needs to be processed on any of the parallel, identical machines from the set $M = \{1, \ldots, m\}$. We consider the time-stamp model of online optimization [19], that is, the jobs arrive over time, every job j at its release date $r_j \geq 0$. Only at this time, the system learns about the existence of a job, and the processing time p_j is revealed. We assume that any job j prefers a lower completion time to a higher one, where the completion time c_j is the moment in time when job j's processing is finished. Each job owns a piece of private information, namely its indifference cost, or weight, which we denote by w_j^*. The weight represents the privately known cost to a job for one additional unit of time spent waiting. We use w_j^* for the indifference cost to differentiate it from the reported weight, w_j, which may be different. We define the *valuation* of job j with indifference cost w_j^* for a schedule that gives it completion time c_j as $-w_j^* c_j$. While jobs behave selfishly trying to maximize their valuations, the social welfare is maximized whenever the weighted sum of completion times $\sum_{j \in J} w_j^* c_j$ is minimum.

3 The MinIncrease Algorithm

Recall our assumption that each of the m machines utilizes the WSPT rule locally, and suppose that each job tries to minimize its completion time with respect to the already present jobs, as it has no information about future job arrivals. Then the following can happen.

Example 1. *Let there be m machines and m jobs with processing times $1,\ 1 - \varepsilon,\ 1 - 2\varepsilon, \ldots, 1 - (m-1)\varepsilon$ for constant ε with $0 < \varepsilon < 1/m$, and assume that each job has unit weights. Let all jobs arrive at time zero, but in the given order. (One could enforce this order by slightly changing the release dates and adding dummy jobs, which would not influence the demonstrated effect, but complicate notation.)*

Then an optimal schedule assigns exactly one job to every machine, resulting in $\sum w_j^* c_j < m$. If each job can select a machine itself, an arriving job only finds jobs already scheduled on machines with a larger processing time. By an inductive argument, no incentives exist to report a false weight, since any arriving job will be the shortest, and therefore any of the m machines would minimize the job's completion time. It is therefore possible that all jobs choose the same machine[1]. The weighted sum of completion times is then larger than $(1-(m-1)\varepsilon)(m(m+1)/2)$ and therefore the approximation ratio is bounded from below by

$$\frac{(1-(m-1)\varepsilon)(m(m+1)/2)}{m} = \frac{m+1}{2} - \frac{(m+1)(m-1)}{2}\,\varepsilon,$$

which becomes arbitrarily large for large m.

Hence, a decentralized selection of machines by the jobs themselves can cause arbitrarily large deviations from the optimum. On the other hand, using central coordination one can enforce the solution to be O(1)-competitive, that is, not worse than a constant times the off-line optimum. We next propose an algorithm that is inspired by the MinIncrease algorithm in [16], which yields a competitive ratio of 3.281. Since we have to rely on reported weights w_j, we consider $\sum_{j \in J} w_j c_j$ instead of $\sum_{j \in J} w_j^* c_j$. In order to formulate the algorithm, we first introduce the necessary notation. Let $c_j(i)$ denote the completion time of job j when assigned to machine i. Let $j \to i$ denote the fact that job j is scheduled on machine i. Without loss of generality, we assume that the jobs are

[1] Again, one could make the first machine the only one maximizing the utility of an arriving job by adding dummy jobs with small processing times that occupy all other machines, before any 'real' job arrives.

numbered in order of their arrival, i.e. $j < k \Rightarrow r_j \leq r_k$. For any job j, let $H(j)$ denote the set of jobs that have higher priority than j according to WSPT, i.e.,

$$H(j) = \left\{k \in J \mid \frac{w_k}{p_k} > \frac{w_j}{p_j}\right\} \cup \left\{k \leq j \mid \frac{w_k}{p_k} = \frac{w_j}{p_j}\right\}.$$

Note that $H(j)$ includes j, too. Similarly, let $L(j) = J \setminus H(j)$ denote the set of jobs with lower priority. In case of equal ratios w_j/p_j, we break ties by giving higher priority to jobs that arrive earlier according to the online sequence. Furthermore, let s_j denote the starting time of j, i.e., the time when j eventually starts being processed. Clearly, $s_j \geq r_j$. At a given point t in time, machine i might be busy processing a job. Let $b_i(t)$ denote the remaining processing time of that job at time t, i.e., at time t machine i will be blocked during $b_i(t)$ units of time for new jobs. If machine i is idle at time t, set $b_i(t) = 0$. The algorithm consists of a local scheduling policy, the WSPT rule, that is applied by every machine and an assignment procedure that is used whenever a new job arrives.

Algorithm 1: MinIncrease Algorithm

Local Sequencing Policy: Whenever a machine becomes idle, it starts processing the job with highest priority among all available jobs assigned to it. Priority here means the ratio of reported weight over processing time. In case of equal ratios a job with smaller index has higher priority.

Assignment:

1. At time r_j job j arrives and reports a weight w_j (possibly $w_j \neq w_j^*$).
2. For every machine $i \in M$ the increase in the objective value (where the true weights w_j^* are replaced by the reports w_j) is computed. The increase of j on machine i is

$$incr(j,i) = w_j c_j(i) + p_j \sum_{\substack{k \in L(j) \\ k \to i \\ k < j \\ s_k > r_j}} w_k = w_j[r_j + b_i(r_j) + \sum_{\substack{k \in H(j) \\ k \to i \\ k < j \\ s_k > r_j}} p_k + p_j] + p_j \sum_{\substack{k \in L(j) \\ k \to i \\ k < j \\ s_k > r_j}} w_k.$$

3. Job j is assigned to machine $i_j \in \operatorname{argmin}_{i \in M} incr(j,i)$ with minimum index.

The MinIncrease Algorithm still makes use of central coordination in Step 3. In the sequel we will first analyze the MinIncrease Algorithm, and then introduce payments that allow a decentralized implementation of the algorithm.

4 The Mechanism Design Perspective

In order to get hands on the quality of a schedule, our aim is to motivate the jobs to report their private piece of information, their indifference costs w_j^*, truthfully. Therefore we give a definition of truthfulness which requires some mechanism design notation first. In mechanism design, one refers to the private information of an agent as its *type*. Let us regard the job set $J = \{1, \ldots, n\}$ as a of agents, each having a true type w_j^* from the space of possible types W. Given a vector of reports $\mathbf{w} = (w_1, \ldots, w_n)$ of all agents (jobs), an *allocation algorithm* $A\colon W^n \to O$ computes an *outcome* $A(\mathbf{w})$ from the set of possible outcomes O. Here, the set of outcomes O is the set of all possible schedules. A *payment rule* $\pi : W^n \to \mathbb{R}^n$ determines payments $\pi_1(\mathbf{w}), \ldots, \pi_n(\mathbf{w})$ for every agent. The tuple $\mu = (A, \pi)$ is called a *mechanism*. We confine ourselves to *direct revelation mechanisms*, where the strategy of each agent j is simply to report a type $w_j \in W$. We assume that agents

have *quasi-linear utilities*, i.e., an agent j's utility is computed from its *valuation* $v_j(A(\mathbf{w})|w_j^*)$ (i.e., its valuation v_j for the outcome $A(\mathbf{w})$, given its true type w_j^*) and its payment $\pi_j(\mathbf{w})$ as follows: $u_j(\mu(\mathbf{w})|w_j^*) = v_j(A(\mathbf{w})|w_j^*) - \pi_j(\mathbf{w})$.

The valuation of a job j for a schedule that gives it completion time c_j is $-w_j^* c_j$. The corresponding utility if j has to pay π_j will be abbreviated by u_j and is therefore $u_j = -w_j^* c_j - \pi_j$. We will deal with non-negative payments $\pi_j \geq 0$ only, i.e., jobs have to pay a non-negative amount for being processed. With this notation, u_j is always negative. Therefore, we assume that a job has a constant and sufficiently large utility for 'being processed at all'. That would add a constant to u_j such that the true utility is always positive. Since this does not change the jobs' behavior when maximizing their utility, we will omit the constant and continue working with u_j.

Definition 2. *A direct revelation mechanism μ is called* truthful *or* dominant strategy incentive compatible *if for all agents $j \in J$, all fixed reports of the other agents $\boldsymbol{w}_{-j} = (w_1, \ldots, w_{j-1}, w_{j+1}, \ldots, w_n)$ and all possible reports $w_j \in W$, $u_j(\mu(\boldsymbol{w}_{-j}, w_j^*)|w_j^*) \geq u_j(\mu(\boldsymbol{w}_{-j}, w_j)|w_j^*)$. That is, if reporting the truth is a weakly dominant strategy for each agent.*

An allocation algorithm that computes outcomes that would be desirable for a social planner is called *social choice function*. The overall goal is to design a mechanism that implements this social choice function. In the model we consider, we assume a *utilitarian* social choice function. This is one of the common goals when maximizing the social welfare [17].

Definition 3. *A social choice function f is called* utilitarian *if it maximizes the sum of valuations of all agents, i.e., $f(\boldsymbol{w}) \in \operatorname{argmax}_{o \in O} \sum_{j \in J} v_j(o|w_j)$.*

The social choice function is thus an algorithm maximizing $\sum_{j \in J} -w_j^* c_j$, or equivalently minimizing $\sum_{j \in J} w_j^* c_j$. Our goal is thus the design of a mechanism that implements this social choice function, i.e., a mechanism that yields a schedule minimizing $\sum_{j \in J} w_j^* c_j$. For utilitarian social choice functions we have the following well known theorem.

Theorem 4. (Groves [7]) *If the allocation algorithm A computes the utilitarian social choice function for every input vector $\boldsymbol{w}$, then there is a payment scheme π such that the direct revelation mechanism (A, π) is truthful.*

In other words, an algorithm that computes an optimal schedule for $\sum w_j c_j$ for any vector of weights $\mathbf{w}$, can be extended to a truthful mechanism with appropriate payments. Scheduling to minimize the weighted sum of completion times with release dates, however, is NP-hard, even in the off-line case [14]. Furthermore, no online algorithm for the single machine problem can be better than 2-competitive [9] regardless of the question whether or not P=NP, and lower bounds exist for parallel machines, too [21]. Therefore, the social choice function cannot be computed due to lack of both unbounded computational resources and information. Moreover, it is known that Theorem 4 does not generalize to the case where an approximation of the social choice function is used, this was shown by Nisan and Ronen [18]. And indeed, it is not possible to find a payment π that completes the MinIncrease Algorithm to a truthful mechanism. To illustrate the latter, we use the following necessary condition formulated by Lavi et al. [13].

Definition 5. (Weak Monotonicity) *An allocation algorithm A satisfies* weak monotonicity *if for any agent $j \in J$, every fixed report vector of the other agents w_{-j} and every pair of possible types $\widetilde{w}_j$ and w_j*

$$v_j(A(w_{-j}, \widetilde{w}_j)|\widetilde{w}_j) - v_j(A(w_{-j}, \widetilde{w}_j)|w_j) \geq v_j(A(w_{-j}, w_j)|\widetilde{w}_j) - v_j(A(w_{-j}, w_j)|w_j).$$

Lemma 6. (Lavi, Mu'alem, and Nisan [13]) *Let A be an allocation algorithm. If there is a payment scheme π such that (A, π) is a truthful mechanism, then A satisfies weak monotonicity.*

This result is now applied to our model. Lemma 7 reformulates weak monotonicity in terms of our valuation functions.

Lemma 7. *For a job $j \in J$ and fixed reports w_{-j} by the other jobs, let $A(w_{-j}, w_j)$ denote the resulting schedule if j reports w_j. Let $c_j(A(w_{-j}, w_j))$ be the corresponding (ex-post) completion time of j in that schedule. Then A satisfies weak monotonicity in the described model if and only if it satisfies*

$$w_j < \widetilde{w}_j \Rightarrow c_j(A(w_{-j}, w_j)) \geq c_j(A(w_{-j}, \widetilde{w}_j))$$
$$\forall\, j \in J, \quad \forall\, w_{-j} \in W^{n-1}, \quad \forall\, w_j, \widetilde{w}_j \in W.$$

Proof. See Appendix A. □

The above condition is in fact equivalent to the notion of decreasing work curves as formulated by Archer and Tardos [2]. An example in Appendix B shows that the MinIncrease Algorithm does not satisfy weak monotonicity, and therefore does not allow a payment scheme that extends the algorithm to a mechanism that makes truth-telling an ex-post weakly dominant strategy (i.e., a truthful mechanism). Let us summarize this.

Theorem 8. *There does not exist a payment scheme that extends the* MinIncrease *algorithm to a truthful mechanism.*

Proof. Use Lemma 7 and the example in Appendix B. □

5 Payments for Myopic Rational Jobs

We cannot extend the MinIncrease algorithm with a payment scheme that makes truth-telling an ex-post dominant strategy. Therefore, we focus on the moment when a job arrives and is assigned to a machine, and propose a payment scheme that makes truth-telling at least a myopic best response for any arriving job. That is, at time t when a job announces its reported weight w_j, truth-telling is a strategy that maximizes the job's utility on the basis of the available information at time t.

The payments we introduce are motivated by the Vickrey Clarke Groves (VCG) mechanism [7]. That is, a job j pays at the moment of its placement on one of the machines an amount that compensates the decrease in utility of the other jobs. Besides making the mechanism truthful (in a myopic sense that is weaker than dominant strategy incentive compatible!), these payments give us the opportunity to decentralize the algorithm. If we let jobs select a machine themselves, myopic rational jobs select the machine that the MinIncrease Algorithm would have selected, too. We will see in the next section that this can be turned into a constant-factor approximation of the off-line optimum, given that the jobs behave rationally. The algorithm including the payments is displayed below as the Decentralized MinIncrease Algorithm.

The Decentralized MinIncrease Algorithm together with the stated payments results in a balanced budget for the scheduler. That is, the payments paid and received by the jobs sum up to zero, since every arriving job immediately makes its payment to the jobs that are displaced by it.

Moreover, although reporting the truth does not necessarily result in an ex-post equilibrium, a truth-telling job is guaranteed the initial utility it achieves when being scheduled at arrival. That is, whenever a job's utility is affected by an arriving job, the decrease in utility caused by an increasing completion time is immediately compensated for by a payment.

Algorithm 2: DECENTRALIZED MININCREASE Mechanism

Local Sequencing Policy: Whenever a machine becomes idle, it starts processing the job with highest priority among all available jobs assigned to it. Priority here means the ratio of reported weight over processing time. In case of equal ratios a job with smaller index has higher priority.

Assignment:

1. At time r_j job j arrives and reports a type w_j (possibly $w_j \neq w_j^*$).
2. For every machine i, job j observes the current situation and computes
$$c_j(i) = r_j + b_i(r_j) + \sum_{\substack{k \in H(j) \\ k \to i \\ k<j \\ s_k > r_j}} p_k + p_j \quad \text{and} \quad \pi_j(i) = p_j \sum_{\substack{k \in L(j) \\ k \to i \\ k<j \\ s_k > r_j}} w_k.$$
3. Job j chooses a machine $i_j \in M$. Its utility for being scheduled on machine i is $u_j(i) = -w_j^* c_j(i) - \pi_i(j)$.
4. The job is scheduled on i_j according to WSPT among all currently available jobs on i_j whose processing has not started yet. The payment $\pi_{i_j}(j)$ has to be paid by j.
5. The completion time for every job $k \in L(j)$, $k \to i_j$, $k < j$, $s_k > r_j$ increases by p_j due to j's presence. As compensation, k receives a payment of $w_k p_j$.

Theorem 9. *Under the* DECENTRALIZED MININCREASE *mechanism, an arriving job maximizes its current utility u_j by reporting its true weight w_j^*. Also ex-post, the job will be left with the same utility u_j. Furthermore, any report $w_j \neq w_j^*$ may lead to a suboptimal utility.*

Proof. We first regard the single machine case, i.e., $m = 1$. Suppose, at the arrival of job j jobs $k_1, k_2, \ldots, k_r$ with corresponding processing times $p_1, p_2, \ldots, p_r$ and reported weights $w_1, w_2, \ldots, w_r$ are queueing to be processed on the machine, but none of them has started being processed yet. Without loss of generality let $w_1/p_1 \geq w_2/p_2 \geq \cdots \geq w_r/p_r$. By choosing its weight appropriately, job j could be scheduled at any position in front of, between or behind the already present jobs. Therefore, it has to decide for every job k_s, $s \in \{1, \ldots, r\}$, whether it wants to be placed in front of k_s or not. Displacing k_s would increase $\pi_j(1)$ by $w_s p_j$, whereas $c_j(1)$ is decreased by p_s. Thus, j's utility changes by $w_j^* p_s - w_s p_j$ if j displaces k_s. Therefore, it is rational for j to displace k_s if and only if $w_j^* p_s - w_s p_j > 0$, which is equivalent to $w_j^*/p_j > w_s/p_s$. As the machine schedules according to WSPT, j is placed at the position that maximizes its utility when reporting w_j^*. Therefore, truth-telling is a dominant strategy in the myopic sense. Note that j can achieve the position between job k_{s-1} and k_s by reporting any $w_j \in (p_j\, w_s/p_s, p_j\, w_{s-1}/p_{s-1}]$. Thus, reporting w_j^* is not the only myopic best response.

For $m > 1$, recall that j can select a machine itself. As reporting the truth maximizes its utility on every single machine, and as j can then choose the machine that maximizes its utility among all machines, truth-telling will maximize j's immediate utility at arrival.

For the second part of the claim, suppose k is displaced by the arriving job j. The current completion time of k thus increases by p_j. Therefore, k's utility decreases by $w_k^* p_j$. If k has reported truthfully it receives a payment of $w_k^* p_j$ in Step 5 of the algorithm, upon j's arrival. Hence, job k immediately receives a payment from job j that exactly compensates for the delay.

For the last claim, suppose j reports $w_j > w_j^*$. Then it may happen that some job k is assigned to the same machine, and $w_j/p_j > w_k/p_k > w_j^*/p_j$. With report w_j, job j would be scheduled in front of k, whereas w_j^* would give it the place behind k. The increase in j's utility when reporting

w_j over its utility for reporting w_j^* is therefore $w_j^* p_k - w_k p_j < 0$. An analogous argument shows that reporting less than w_j^* can be non-optimal, too. □

6 Performance of the mechanism

It is not a goal in itself to have a truthful mechanism, but to use the truthfulness in order to achieve a reasonable overall performance in terms of the social welfare $\sum w_j^* c_j$. The DECENTRALIZED MININCREASE Algorithm as stated above, however, does not yet yield a constant approximation factor; simple examples can be constructed in the same flavor as in [15]. In fact, it can be considered folklore that early arriving jobs with large processing times are critical and have to be delayed [1, 15, 16]. In order to achieve a constant competitive ratio, we also adopt this idea and use modified release dates as [15, 16]. To this end, we change the release date of every job $j \in J$ from r_j to $r'_j = \max\{r_j, \alpha p_j\}$, where α is a constant that will later be chosen appropriately. That is, a job is considered critical if the job's original release date r_j is smaller than α times its processing time p_j. A critical job will be ignored until time $\alpha p_j > r_j$. Only then it has to select a machine and report a weight w_j. Note that the aforementioned properties concerning the balanced budget and the conservation of utility still apply to the algorithm with modified release dates. According to Theorem 9, rational jobs report their weights truthfully. The theorem still applies to the case with modified release dates, as the arguments in the proof of Theorem 9 do not depend on the actual time when the job has to decide (and report a w_j). Yet, it has to be noted that the modification of release dates restricts the level of decentralization in the resulting mechanism, as some central control has to assure that a critical job cannot advance to a machine between its true and its modified release date. We obtain the following theorem.

Theorem 10. *Suppose every job is rational in reporting a weight w_j and selecting a machine. Then the* DECENTRALIZED MININCREASE *algorithm, together with the modified release dates $r'_j = \max\{r_j, \alpha p_j\}$ for critical jobs, is ϱ-competitive, with $\varrho = 3.281$.*

Proof. First, according to Theorem 9, we can assume that jobs report their weights w_j^* truthfully, and moreover, select a machine that minimizes their utility. That is, they select a machine i that minimizes

$$u_j(i) = w_j^* c_j(i) + \pi_i(j) = w_j^* \Big[r'_j + b_i(r_j) + \sum_{\substack{k \in H(j) \\ k \to i \\ k < j \\ s_k > r'_j}} p_k + p_j \Big] + p_j \sum_{\substack{k \in L(j) \\ k \to i \\ k < j \\ s_k > r'_j}} w_k^* .$$

This, however, exactly equals the immediate increase of the objective value $\sum w_j^* c_j$ that is due to the addition of job j to the schedule. We now claim that we can express the objective value Z of the resulting schedule as

$$Z = \sum_{j \in J} u_j(i_j) \,,$$

where i_j is the machine selected by job j. Here, it is important to note that $u_j(i_j)$ does not express the total (ex-post) contribution of job j to $\sum w_j^* c_j$, but only the increase *upon arrival* of j on machine i_j. However, further contributions of job j to $\sum w_j^* c_j$ only appear when job j is displaced by some later arriving job with higher priority, say k. This contribution by job j to $\sum w_j^* c_j$, however, will be accounted for when adding $u_k(i_k)$.

Next, since we assume that any job maximizes its utility upon arrival, or equivalently minimizes $u_j(i)$ when selecting a machine i, we can apply an averaging argument over the number of machines, like in [16], to obtain:

$$Z \leq \sum_{i \in J} \frac{1}{m} \sum_{i=1}^{m} u_j(i) \,.$$

The remainder of the proof utilizes the definitions of $u_j(i)$ and particulary the fact that, upon arrival of job j on any of the machines i (at time r'_j), machine i is blocked for time $b_i(r'_j)$, which is upper bounded by r'_j/α. This upper bound is machine-independent, and follows from the definition of r'_j, since any job k in process at time r'_j fulfills $\alpha p_k \leq r'_k \leq r'_j$. Furthermore, the proof utilizes a lower bound on any (off-line) optimum schedule from Eastman et al. [5, Thm. 1]. The details are moved to Appendix C, due to space limitations. The resulting performance bound 3.281 is identical to the one of [15], letting α equal $(\sqrt{17m^2 - 2m + 1} - m + 1)/(4m)$. □

7 Discussion and Extensions

In the proposed model for decentralized online scheduling, our assumption was that any job experiences costs w_j^* for waiting one extra unit of time, known only to the job itself. Moreover, we demand a certain level of decentralization by letting the arriving jobs choose their machines themselves. The social welfare is maximal when the total weighted completion time $\sum w_j\, c_j$ is minimal. There are three drawbacks of the proposed mechanism that we briefly discuss next.

The jobs truthfully report their weights w_j^* under the assumption of myopic rational behavior. A stronger result would be that truth-telling is an ex-post dominant strategy. This can indeed be achieved when giving up the requirement that jobs choose the machines themselves. For example, when jobs are distributed over machines uniformly at random, utilizing a characterization of Gui et al. [8], we can determine a payment scheme in polynomial time such that the resulting mechanism is dominant strategy incentive compatible, independent of the realization of the random choices. The performance bound remains 3.281 (in expectation). However, when giving up on the decentralization, one can as well set up a dominant strategy incentive compatible mechanism that is based upon a recently proposed algorithm by Correa and Wagner [4], yielding a (deterministic) competitive ratio of 2.62. But notice that, in both cases, the resulting payments can only be determined ex-post, and cannot be implemented online, like with the mechanism we propose.

As a tribute to the desired constant competitive ratio, we had to take special care of critical jobs (i.e., jobs with $\alpha p_j > r_j$), delaying their moment of decision from r_j to $r'_j = \alpha\, p_j$. Clearly, as stated already above, this restricts the level of decentralization, as some central control has to implement this delay of critical jobs. However, given the restriction that jobs choose machines themselves, we are not aware of a better way to handle critical jobs.

We argued that the social welfare is maximized whenever the total weighted completion time of jobs $\sum w_j\, c_j$ is minimal. Given that we have to confine ourselves with approximations, it would be more desirable, though, to minimize the total weighted flow time, $\sum w_j(c_j - r_j)$. However, note that for a single machine and unit weights, this problem does not even admit a constant-factor approximation algorithm in the off-line setting [10], unless P=NP. Nevertheless, it would be an interesting direction for future research to consider also other metrics than $\sum w_j\, c_j$, e.g. stretch metrics as proposed in [3].

Acknowledgements. Thanks to Dries Vermeulen for some helpful discussions.

References

[1] E. J. Anderson and C. N. Potts. Online scheduling of a single machine to minimize total weighted completion time. *Mathematics of Operations Research*, 29(3):686–697, August 2004.

[2] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. 42nd Annual Symposium on Foundations of Computer Science*, pages 482–491. IEEE Computer Society, 2001.

[3] M. A. Bender, S. Chakrabarti, and S. Muthukrishnan. Flow and stretch metrics for scheduling continuous job streams. In *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 270–279. ACM-SIAM, 1998.

[4] J. R. Correa and M. R. Wagner. LP-based online scheduling: from single to parallel machines. In M. Jünger and V. Kaibel, editors, *Proc. 11th Conference on Integer Programming and Combinatorial Optimization*, volume 3509 of *Lecture Notes in Computer Science*, pages 196–209. Springer, 2005.

[5] W. L. Eastman, S. Even, and I. M. Isaacs. Bounds for the optimal scheduling of n jobs on m processors. *Management Science*, 11:268–279, 1964.

[6] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[7] T. Groves. Incentives in teams. *Econometrica*, pages 617–631, 1973.

[8] H. Gui, R. Müller, and R. Vohra. Dominant strategy mechanisms with multidimensional types. Discussion paper, The Center for Mathematical Studies in Economics & Management Sciences, Northwestern University, Evanston, IL, October 2004.

[9] J. A. Hoogeveen and A. P. A. Vestjens. Optimal on-line algorithms for single machine scheduling. In W. H. Cunningham, S. T. McCormick, and M. Queyranne, editors, *Proc. 5th Conference on Integer Programming and Combinatorial Optimization*, volume 1084 of *Lecture Notes in Computer Science*, pages 404–414. Springer, 1996.

[10] H. Kellerer, T. Tautenhahn, and G. J. Woeginger. Approximability and nonapproximability results for minimizing total flow time on a single machine. *SIAM Journal on Computing*, 28:1155–1166, 1999.

[11] E. Koutsoupias and C. Papdimitriou. Worst-case equilibria. In C. Meinel and S. Tison, editors, *Proc. 16th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 404–413. Springer, 1999.

[12] A. Kovacs. Fast monotone 3-approximation algorithm for scheduling related machines. In G. S. Brodal and S. Leonardi, editors, *Proc. 13th Annual European Symposium on Algorithms*, Lecture Notes in Computer Science. Springer, 2005, to appear.

[13] R. Lavi, A. Mu'alem, and N. Nisan. Towards a characterization of truthful combinatorial auctions. In *Proc. 44th Annual Symposium on Foundations of Computer Science*, pages 574–583. IEEE Computer Society, 2003.

[14] E. L. Lenstra, A. H. G. Rinooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:243–362, 1977.

[15] N. Megow and A. S. Schulz. On-line scheduling to minimize average completion time revisited. *Operations Research Letters*, 32:485–490, 2004.

[16] N. Megow, M. Uetz, and T. Vredeveld. Stochastic online scheduling on parallel machines. In G. Persiano and R. Solis-Oba, editors, *Proc. Second International Workshop on Approximation and Online Algorithms*, volume 3351 of *Lecture Notes in Computer Science*, pages 167–180. Springer, 2005.

[17] H. Moulin. *Axioms of Cooperative Decision Making.* Cambridge University Press, Cambridge, USA, 1988.

[18] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[19] K. Pruhs, J. Sgall, and E. Torng. Online scheduling. In J. Y-T. Leung, editor, *Handbook of Scheduling*, chapter 15. CRC Press LLC, 2004.

[20] W. Smith. Various optimizers for single stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.

[21] A. P. A. Vestjens. *On-line Machine Scheduling*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 1997.

[22] M. P. Wellman, W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, to appear.

Appendices

A Proof of Lemma 7

Lemma 7. *Let for a job $j \in J$ and fixed reports w_{-j} for the other jobs $A(w_{-j}, w_j)$ denote the resulting schedule if j reports w_j as computed by allocation algorithm A. Let $c_j(A(w_{-j}, w_j))$ be the corresponding (ex-post) completion time of j in that schedule. Then A satisfies weak monotonicity in our described model if and only if it satisfies*

$$w_j < \widetilde{w}_j \Rightarrow c_j(A(w_{-j}, w_j)) \geq c_j(A(w_{-j}, \widetilde{w}_j))$$
$$\forall\, j \in J\,, \quad \forall\, w_{-j} \in W^{n-1}\,, \quad \forall\, w_j, \widetilde{w}_j \in W.$$

Proof. Let $w_j < \widetilde{w}_j$. Using the special structure of the valuation function in our model the weak monotonicity condition becomes:

$$\begin{aligned}
& v_j(A(w_{-j}, \widetilde{w}_j)|\widetilde{w}_j) - v_j(A(w_{-j}, \widetilde{w}_j)|w_j) - v_j(A(w_{-j}, w_j)|\widetilde{w}_j) + v_j(A(w_{-j}, w_j)|w_j) \geq \quad 0 \\
\Leftrightarrow & \\
& -\widetilde{w}_j c_j(A(w_{-j}, \widetilde{w}_j)) + w_j c_j(A(w_{-j}, \widetilde{w}_j)) + \widetilde{w}_j c_j(A(w_{-j}, w_j)) - w_j c_j(A(w_{-j}, w_j)) \geq \quad 0 \\
\Leftrightarrow & \\
& (\widetilde{w}_j - w_j)[c_j(A(w_{-j}, w_j)) - c_j(A(w_{-j}, \widetilde{w}_j))] \geq \quad 0 \\
\Leftrightarrow & \\
& c_j(A(w_{-j}, w_j)) - c_j(A(w_{-j}, \widetilde{w}_j)) \geq \quad 0\,,
\end{aligned}$$

where the last equivalence follows from $w_j < \widetilde{w}_j$. □

B The MinIncrease algorithm is not weakly monotone

Example 11. *Let $[w/p]$ denote a job with (reported) weight w and processing time p. Suppose that we have to schedule the following four jobs on two machines: $[6/3], [5/4], j = [w/\frac{1}{7}], [20/4]$, where w is a parameter. Let all jobs have release date zero, but let us assume that they nevertheless arrive in the given order. (We could alternatively enforce this order by adding small but positive constants to some of the release dates without changing the effect demonstrated below.)*

Let us consider the MinIncrease algorithm. The first job $[6/3]$ increases the objective value on both machines by the same amount and is therefore scheduled on the first machine. The second job $[5/4]$ is then assigned to the second machine. We consider two values for the weight of j, namely $w^1 = \frac{1}{14}$ and $w^2 = \frac{1}{2}$. In the first case the weight over processing time ratio is $\frac{1}{2}$ and therefore smaller than the respective ratios of the two jobs already assigned to machines. Thus, j would be scheduled last on each of the machines according to the WSPT rule. It would cause the following increases:

$$incr(j, 1) = \frac{1}{7}w^1 + 3w^1$$

$$incr(j,2) = \frac{1}{7}w^1 + 4w^1.$$

Therefore, j is assigned to the end of machine 1, which results in the preliminary schedule depicted on the left of Figure 1.

The second case for $w^2 = \frac{1}{2}$ yields a ratio of $\frac{7}{2}$, which would place j first on both machines. The respective increases are:

$$incr(j,1) = \frac{1}{7}w^2 + 6 \cdot \frac{1}{7}$$
$$incr(j,2) = \frac{1}{7}w^2 + 5 \cdot \frac{1}{7}.$$

Job j would be scheduled on machine 2 yielding the schedule depicted on the right of Figure 1. The last

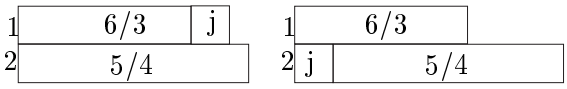

Figure 1: Schedules for Example 11

job [20/4] has a ratio larger than all the ratios of the present jobs. Therefore it would be scheduled first on both machines. In both cases the total weight of jobs on the first machine is larger than the total weight of jobs on the second machine. Therefore the increase in the objective value caused by the last job is in both cases smaller on the second machine. Thus the job is scheduled on the second machine, which increases j's completion time only in the second case. Thus, j is completed at time $3 + \frac{1}{7}$ when reporting $\frac{1}{14}$ and at time $4 + \frac{1}{7}$ when reporting $\frac{1}{2}$. Therefore, the MinIncrease algorithm does not satisfy weak monotonicity.

C Proof of Theorem 10

Theorem 10. *Suppose every job is rational in reporting a weight w_j and selecting a machine. Then the* Decentralized MinIncrease *algorithm, together with the modified release dates $r'_j = \max\{r_j, \alpha p_j\}$ for critical jobs, is ϱ-competitive, with $\varrho = 3.281$.*

Proof. Recall that Z denotes the objective value of the final schedule produced by the Decentralized MinIncrease algorithm. Let Z^{OPT} denote the value of the optimum off-line solution. We have already argued that

$$Z \le \sum_{i \in J} \frac{1}{m} \sum_{i=1}^{m} u_j(i)\,.$$

Next, recall that upon arrival of job j on any of the machines i (at time r'_j), machine i is blocked for time $b_i(r'_j) \le r'_j/\alpha$. Therefore we get, for any j,

$$\begin{aligned}
\frac{1}{m}\sum_{i=1}^{m} u_j(i) &= w_j^* r'_j + w_j^* \sum_{i=1}^{m} \frac{b_i(r'_j)}{m} + w_j^* \sum_{i=1}^{m} \sum_{\substack{k \in H(j) \\ k \to i \\ k<j \\ s_k > r'_j}} \frac{p_k}{m} + w_j^* p_j + p_j \sum_{i=1}^{m} \sum_{\substack{k \in L(j) \\ k \to i \\ k<j \\ s_k > r'_j}} \frac{w_k^*}{m} \\
&= w_j^* r'_j + w_j^* \sum_{i=1}^{m} \frac{b_i(r'_j)}{m} + w_j^* \sum_{\substack{k \in H(j) \\ k<j \\ s_k > r'_j}} \frac{p_k}{m} + w_j^* p_j + p_j \sum_{\substack{k \in L(j) \\ k<j \\ s_k > r'_j}} \frac{w_k^*}{m} \\
&\le w_j^* r'_j + w_j^* \sum_{i=1}^{m} \frac{b_i(r'_j)}{m} + w_j^* \sum_{\substack{k \in H(j) \\ k<j}} \frac{p_k}{m} + w_j^* p_j + p_j \sum_{\substack{k \in L(j) \\ k<j}} \frac{w_k^*}{m} \\
&\le w_j^* r'_j + w_j^* \frac{r'_j}{\alpha} + w_j^* \sum_{\substack{k \in H(j) \\ k<j}} \frac{p_k}{m} + w_j^* p_j + p_j \sum_{\substack{k \in L(j) \\ k<j}} \frac{w_k^*}{m}\,.
\end{aligned}$$

The last term can be rewritten as follows:

$$\sum_{j\in J} p_j \sum_{\substack{k\in L(j)\\ k<j}} \frac{w_k^*}{m} = \sum_{\substack{(j,k):\\ j\in H(k)\\ k<j}} p_j \frac{w_k^*}{m} = \sum_{\substack{(j,k):\\ k\in H(j)\\ j<k}} p_k \frac{w_j^*}{m} = \sum_{j\in J} w_j^* \sum_{\substack{k\in H(j)\\ k>j}} \frac{p_k}{m}\,.$$

Thus,

$$\begin{aligned} Z &\le \sum_{j\in J} w_j^*(1+\frac{1}{\alpha})r_j' + \sum_{j\in J} w_j^* \sum_{\substack{k\in H(j)\\ k<j}} \frac{p_k}{m} + \sum_{j\in J} w_j^* p_j + \sum_{j\in J} w_j^* \sum_{\substack{k\in H(j)\\ k>j}} \frac{p_k}{m} \\ &= \sum_{j\in J} w_j^*(1+\frac{1}{\alpha})r_j' + \sum_{j\in J} w_j^* \sum_{k\in H(j)} \frac{p_k}{m} + \frac{m-1}{m}\sum_{j\in J} w_j^* p_j\,. \end{aligned}$$

Now, we apply a lower bound on the optimal off-line schedule from [5, Thm. 1], namely

$$Z^{OPT} \ge \sum_{j\in J} w_j^* \sum_{k\in H(j)} \frac{p_k}{m} + \frac{m-1}{2m}\sum_{j\in J} w_j^* p_j\,,$$

yielding:

$$\begin{aligned} Z &\le Z^{OPT} + \sum_{j\in J} w_j^*(1+\frac{1}{\alpha})r_j' + \frac{m-1}{2m}\sum_{j\in J} w_j^* p_j \\ &\le Z^{OPT} + \sum_{j\in J} w_j^*(1+\frac{1}{\alpha})(r_j+\alpha p_j) + \frac{m-1}{2m}\sum_{j\in J} w_j^* p_j \\ &= Z^{OPT} + \sum_{j\in J} w_j^* \left[(1+\frac{1}{\alpha})r_j + (1+\alpha+\frac{m-1}{2m})p_j\right], \end{aligned}$$

where in the second inequality $r_j + \alpha p_j$ is used as an upper bound on r_j'. Applying the trivial lower bound $\sum_{j\in J} w_j^*(r_j+p_j) \le Z^{OPT}$, we get:

$$\begin{aligned} Z &\le Z^{OPT} + \max\left\{1+\frac{1}{\alpha},\, 1+\alpha+\frac{m-1}{2m}\right\} Z^{OPT} \\ &= 2Z^{OPT} + \max\left\{\frac{1}{\alpha},\, \alpha+\frac{m-1}{2m}\right\} Z^{OPT}. \end{aligned}$$

Therefore, we get the performance bound

$$\varrho = 2 + \max\left\{\frac{1}{\alpha},\, \alpha+\frac{m-1}{2m}\right\}.$$

This can now be optimized for α, which was already done in [15]. There it was shown that $\varrho < 3.281$ for $\alpha = \sqrt{17m^2-2m+1} - m + 1)/(4m)$. □